LEVEL II

(1)

AD A102226

DTIC
ELECTE
S JUL 30 1981 D
E

(6)

SCHEDULING WITH SLACK TIME,

(10) Chung Laung ~~C.L.~~ LIU, JANE W.S. LIU, ARTHUR I. LIESTMAN

Department of Computer Science, University of Illinois

Urbana, Illinois. 61801

USA

(11) 1979

(12) 14

This work was partially supported by the US Office of Naval Research under the Contract Number N00014-79-C-0775 ~~and by USARSG~~ DAJA37-79-M-0204

(15)

176 011 ✓

81 7 30 05

1. Introduction

We study in this paper a problem concerning the scheduling of a set of jobs on a single processor computer system. In our model, a job consists of a periodic stream of identical requests. That is, a job J_i demands periodically C_i units of computation time in every T_i units of time. We shall use $J_1, J_2, \ldots, J_n$ to denote the jobs, $T_1, T_2, \ldots T_n$ to denote the request periods, and $C_1, C_2, \ldots, C_n$ to denote the computation times. (Throughout our discussion, we shall assume $T_1 \leq T_2 \leq \ldots \leq T_n$ for convenience in notations). That is, a job J_i is completely characterized by an ordered pair of numbers (C_i, T_i). $< \delta =$

By scheduling a set of jobs we mean to determine, at any time instant, the particular request to which the processor should be devoted. In our model, the criterion of scheduling is to satisfy each request prior to its deadline, which is the arrival time of the next request of the same job. A set of jobs is said to be <u>schedulable</u> by a certain scheduling algorithm if such a criterion can be met. Throughout our dicussion, we assume that the preemptive scheduling discipline is employed. That is, the execution of a job can be interrupted by the execution of another job. Thus, a request of C_i units of computation time can be satisfied by one or more quanta of time which sum to C_i.

A well-known scheduling algorithm for our model is the <u>earliest deadline first</u> scheduling algorithm [1, 2, 3]. In this case, priorities are assigned to the requests according to their deadlines, with highest priority assigned to the request with the earliest deadline. The arrival of a request with a higher priority always preempts the execution of a request with a lower priority. The earliest deadline first scheduling algorithm has been shown to be optimum in the sense that if a set of jobs is schedulable by any scheduling algorithm then the set is also schedulable by the earliest deadline first scheduling algorithm.

Another scheduling algorithm which was studied in [3] in the <u>rate monotonic</u> scheduling algorithm. In this case, higher priority is assigned to a request with shorter request period. Thus, for example, the requests of the job with the shortest request period will always preempt any request of another job. This algorithm is inferior to the earliest deadline first algorithm in the sense that there are sets of jobs which are schedulable by the earliest deadline first algorithm

A

while not schedulable by the rate monotonic algorithm. However, the rate monotonic algorithm is simpler to implement. Furthermore, the simplicity of this algorithm enables us to estimate the slack time of a request, which is the subject of this paper. It is often the case that instead of simply meeting the deadline of each request, we might wish to satisfy each request ahead of its deadline so that there will be a time span between the completion of the execution of a request and the deadline. We call such a time span the slack time of the request.

We summarize here some of the results in [3] that will be useful in our study. We use $u_1, u_2, \ldots u_n$ to denote the ratios $C_1/T_1, C_2/T_2, \ldots, C_n/T_n$. We use u to denote the sum $\sum_{i=1}^{n} C_i/T_i$, u is referred to as the utilization factor of the set of jobs $J_1, J_2, \ldots, J_n$. A set of jobs is said to fully utilize the processor with respect to a particular scheduling algorithm if the set of jobs is schedulable by the algorithm while increasing the computation time of any of the jobs will cause the set to become unschedulable. (Note that the notion of full utilization is defined with respect to a particular scheduling algorithm. For example, the set of jobs $\{J_1 = (1, 2), J_2 = (2,5)\}$ fully utilizes the processor with respect to the rate monotonic scheduling algorithm. However, this set does not fully utilize the processor with respect to the earliest dealine first scheduling algorithm). Proofs of the following theorems can be found in [3] :

Theorem 1 : A set of n jobs with a utilization factor less than $n(2^{1/n} - 1)$ is always schedulable by the rate monotonic scheduling algorithm.

Theorem 2 : A set of n jobs with a utilization factor less than $n(2^{1/n} - 1)$ does not fully utilize the processor with respect to the rate monotonic scheduling algorithm. Moreover, there exists a set of n jobs with a utilization factor equal to $n(2^{1/n} - 1)$ that fully utilizes the processor.

while not schedulable by the rate monotonic algorithm. However, the rate monotonic algorithm is simpler to implement. Furthermore, the simplicity of this algorithm enables us to estimate the <u>slack time</u> of a request, which is the subject of this paper. It is often the case that instead of simply meeting the deadline of each request, we might wish to satisfy each request ahead of its deadline so that there will be a time span between the completion of the execution of a request and the deadline. We call such a time span the <u>slack time</u> of the request.

We summarize here some of the results in [3] that will be useful in our study. We use $u_1, u_2, \ldots u_n$ to denote the ratios $C_1/T_1, C_2/T_2, \ldots, C_n/T_n$. We use u to denote the sum $\sum_{i=1}^{n} C_i/T_i$. u is referred to as the <u>utilization factor</u> of the set of jobs $J_1, J_2, \ldots, J_n$. A set of jobs is said to <u>fully utilize</u> the processor with respect to a particular scheduling algorithm if the set of jobs is schedulable by the algorithm while increasing the computation time of any of the jobs will cause the set to become unschedulable. (Note that the notion of full utilization is defined with respect to a particular scheduling algorithm. For example, the set of jobs $\{J_1 = (1, 2), J_2 = (2,5)\}$ fully utilizes the processor with respect to the rate monotonic scheduling algorithm. However, this set does not fully utilize the processor with respect to the earliest dealine first scheduling algorithm). Proofs of the following theorems can be found in [3] :

<u>Theorem 1</u> : A set of n jobs with a utilization factor less than $n(2^{1/n} - 1)$ is always schedulable by the rate monotonic scheduling algorithm.

<u>Theorem 2</u> : A set of n jobs with a utilization factor less than $n(2^{1/n} - 1)$ does not fully utilize the processor with respect to the rate monotonic scheduling algorithm. Moreover, there exists a set of n jobs with a utilization factor equal to $n(2^{1/n} - 1)$ that fully utilizes the processor.

2. A Fundamental theorem.

Suppose that $J_1, J_2, \ldots, J_n$ are to be scheduled by the rate monotonic scheduling algorithm. We shall assume that the first request of all these jobs arrive at $t=0$. (As a matter of fact, Theorem 3 below shows that such an assumption covers the worst possible case). We define the availability function $f(t)$ to be :

$$f(t) = \begin{cases} 1 & \text{if the processor is occupied at } t \\ 0 & \text{if the processor is not occupied at } t \end{cases}$$

For any Δ and τ, the integral

$$\int_{\Delta}^{\Delta + \tau} f(t)dt$$

gives the total units of time that the processor is not occupied between Δ and $\Delta + \tau$. A sequence of demands (not necessarily periodic) within the time interval $[0, t_k]$ can be denoted

$$D = (d_1, [0, t_1]), (d_2, [t_1, t_2]), \ldots, (d_k, [t_{k-1}, t_k])$$

to mean that d_1 units of computation time is demanded within the time interval $[0, t_1]$, d_2 units of computation time is demanded within the time interval $[t_1, t_2]$, and so on. We shall use D^{Δ} to denote the sequence of demands.

$$D^{\Delta} = (d_1, [\Delta, t_1 + \Delta]), \quad (d_2, [t_1 + \Delta, t_2 + \Delta]), \ldots, (d_k, [t_{k-1} + \Delta, t_k + \Delta])$$

which is D delayed by Δ. Given an availability function $f(t)$ and a sequence of demands D within the time interval $[0, t_k]$, and assumed that the demands in D have priorities lower than all the requests of the jobs $J_1, J_2, \ldots, J_n$, then the demands in D will take up the first d_1 units of time within the time interval $[0, t_1]$ in $f(t)$, and so on. (Assume that $f(t)$ can satisfy the demands in D).
We shall use

$$f(t) - D(t)$$

to denote the available time that remains after all the demands in D are satisfied. (This is a slight abuse of notation, since straightly speaking a sequence of demands is not a function of time.
Rather, D induces a corresponding $D(t)$ for a given $f(t)$, and will induce a different $D(t)$ for a different $f(t)$.) The following Lemmas are obvious :

2. A Fundamental theorem.

Suppose that J_1, J_2, ..., J_n are to be scheduled by the rate monotonic scheduling algorithm. We shall assume that the first request of all these jobs arrive at $t=0$. (As a matter of fact, Theorem 3 below shows that such an assumption covers the worst possible case). We define the availability function $f(t)$ to be :

$$f(t) = \begin{cases} 1 & \text{if the processor is occupied at } t \\ 0 & \text{if the processor is not occupied at } t \end{cases}$$

For any Δ and τ, the integral

$$\int_{\Delta}^{\Delta+\tau} f(t)dt$$

gives the total units of time that the processor is not occupied between Δ and $\Delta + \tau$. A sequence of demands (not necessarily periodic) within the time interval $[0, t_k]$ can be denoted

$$D = (d_1, [0, t_1]), (d_2, [t_1, t_2]), \ldots, (d_k, [t_{k-1}, t_k])$$

to mean that d_1 units of computation time is demanded within the time interval $[0, t_1]$, d_2 units of computation time is demanded within the time interval $[t_1, t_2]$, and so on. We shall use D^{Δ} to denote the sequence of demands.

$$D^{\Delta} = (d_1, [\Delta, t_1 + \Delta]), (d_2, [t_1 + \Delta, t_2 + \Delta]), \ldots, (d_k, [t_{k-1} + \Delta, t_k + \Delta])$$

which is D delayed by Δ. Given an availability function $f(t)$ and a sequence of demands D within the time interval $[0, t_k]$, and assumed that the demands in D have priorities lower than all the requests of the jobs J_1, J_2, ..., J_n, then the demands in D will take up the first d_1 units of time within the time interval $[0, t_1]$ in $f(t)$, and so on. (Assume that $f(t)$ can satisfy the demands in D).
We shall use

$$f(t) - D(t)$$

to denote the available time that remains after all the demands in D are satisfied. (This is a slight abuse of notation, since straightly speaking a sequence of demands is not a function of time.
Rather, D induces a corresponding $D(t)$ for a given $f(t)$, and will induce a different $D(t)$ for a different $f(t)$.) The following Lemmas are obvious :

Lemma 1: For any $f(t)$ and any D within the time interval $[0, \tau]$, if

$$\int_{\Delta_1}^{\Delta_1 + \tau} f(t)dt \leq \int_{\Delta_2}^{\Delta_2 + \tau} f(t)dt$$

than

$$\int_{\Delta_1}^{\Delta_1 + \tau} [f(t) - D^{\Delta_1}(t)]dt \leq \int_{\Delta_2}^{\Delta_2 + \tau} [f(t) - D^{\Delta_2}(t)]dt$$

Lemma 2 : Let

$D = (d_1, [0,t_1]), (d_2, [t_1, t_2]), \ldots, (d_k, [t_{k-1}, t_k])$

be a sequence of demands. Let

$E = (0, [0, \delta]), (d_1, [\delta, t_1 + \delta]), (d_2, [t_1 + \delta, t_2 + \delta]),$

$\ldots (d_k, [t_{k-1} + \delta, t_k + \delta])$

That is, E is the sequence of demands in D delayed by δ units of time. Then, for any τ

$$\int_{\Delta}^{\Delta + \tau} [f(t) - D^{\Delta}(t)]dt \leq \int_{\Delta}^{\Delta + \tau} [f(t) - E^{\Delta}(t)]dt$$

We prove now a fundamental theorem which will be needed in our later discussion :

Theorem 3 : For any set of n jobs $J_1, J_2, \ldots J_n$ scheduled by the rate monotonic scheduling algorithm, we have

$$\int_0^{\tau} f(t)dt \leq \int_{\Delta}^{\Delta + \tau} f(t)dt$$

for any Δ and τ .

Proof : The theorem is proved by induction on n. As the basis of induction, we note that for $n = 1$, $f(t)$ is a periodic function of period T_1 so that $f(t) = 0$ for the first C_1 units of time and $f(t) = 1$ for the next $T_1 - C_1$ units of time in each period. Clearly,

$$\int_0^{\tau} f(t)dt \leq \int_{\Delta}^{\Delta + \tau} f(t)dt$$

To carry out the induction step, we assume that the theorem is true when $f(t)$ is the availability function after n-1 jobs have been scheduled. Let $\hat{f}(t)$ denote the availability function after n jobs have been scheduled.

Consider first the simple case thatΔ is a multiple of T_n. Let

$$D = (C_n, [0, T_n]), (C_n, [T_n, 2T_n]), \ldots, (C_n, [(k-1)T_n, kT_n])$$

such that $kT_n \geq \tau$. According to the induction hypothesis

$$\int_0^{\tau} f(t)dt \leq \int_{\Delta}^{\Delta + \tau} f(t)dt$$

According to Lemma 1

$$\int_0^{\tau} [f(t) - D(t)]dt \leq \int_{\Delta}^{\Delta + \tau} [f(t) - D^{\Delta}(t)]dt$$

which is

$$\int_0^{\tau} \hat{f}(t)dt \leq \int_{\Delta}^{\Delta + \tau} \hat{f}(t)dt$$

Now consider the case that Δ is not a multiple of T_n as illustrated in Fig 1.

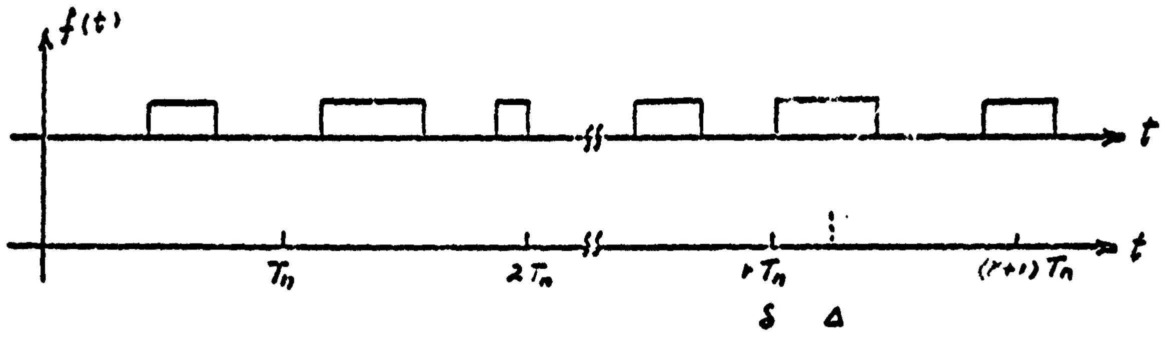


Fig. 1.

Let τ be the largest t such that $\delta < \Delta$ and δ is a multiple of T_n, $\delta = rT_n$.

We examine two cases :

<u>Case 1</u> : $\hat{f}(t) = 0$ for $\delta \leq t \leq \Delta$. In this case

$$\int_{\delta}^{\delta + \tau} \hat{f}(t)dt \leq \int_{\Delta}^{\Delta + \tau} \hat{f}(t)dt \qquad (1)$$

Assording to the induction hypothesis

$$\int_0^{\tau} f(t)dt \leq \int_{\delta}^{\delta + \tau} f(t)dt$$

According to Lemma 1

$$\int_0^{\tau} [f(t) - D(t)]dt \le \int_{\delta}^{\delta+\tau} [f(t) - D^{\delta}(t)]dt$$

That is,

$$\int_0^{\tau} \hat{f}(t)dt \le \int_{\delta}^{\delta+\tau} \hat{f}(t)dt \qquad (2)$$

Combining (1) and (2), we obtain

$$\int_0^{\tau} \hat{f}(t)dt \le \int_{\Delta}^{\Delta+\tau} \hat{f}(t)dt$$

<u>Case 2</u> :

$\hat{f}(t) \neq 0$ for $\delta \le \_ \le \Delta$. That $\hat{f}(t) \neq 0$ implies the demand within the time interval $[rT_n, (r+1)T_n]$ is satisfied at or prior to $t = \Delta$. Thus, within the time interval $[\Delta, (r+1)T_n]$, $\hat{f}(t) = f(t)$.

Let λ denote $(r+1)T_n - \Delta$. Let

$$E = (0, [0, \lambda]), (C_n, [T_n + \lambda, 2T_n + \lambda]), \ldots, (C_n, [kT_n + \lambda, (k+1)T_n + \lambda])$$

That is, E is D delayed by λ. Thus, we can write

$$\int_{\Delta}^{\Delta+\tau} \hat{f}(t)dt = \int_{\Delta}^{\Delta+\tau} [f(t) - E^{\Delta}(t)]dt \qquad (3)$$

According to Lemma 2

$$\int_{\Delta}^{\Delta+\tau} [f(t) - D^{\Delta}(t)]dt \le \int_{\Delta}^{\Delta+\tau} [f(t) - E^{\Delta}(t)]dt \qquad (4)$$

However, according to Lemma 1 and the induction hypothesis

$$\int_0^{\tau} \hat{f}(t)dt \le \int_{\Delta}^{\Delta+\tau} [f(t) - D^{\Delta}(t)]dt \qquad (5)$$

Combining (3), (4), and (5), we obtain

$$\int_0^{\tau} \hat{f}(t)dt \le \int_{\Delta}^{\Delta+\tau} [f(t) - E^{\Delta}(t)]dt = \int_{\Delta}^{\Delta+\tau} \hat{f}(t)dt$$

3. Estimation of slack Time.

We show in this section lower bounds on the slack time of a request. Theorem 4 and 5 in the followings will be proved by induction on the number of jobs. We show first a lemma which be used as the basis of induction, in the proofs of Theorems 4 and 5.

Lemma 3 : Let J_1 and J_2 be two jobs scheduled according to the rate monotonic scheduling algorithm, with $u < 2(2^{1/2}-1)$. For an arbitrary request of J_2, let q denote the size of the last quantum of time allocated to the request and s denote the length of the slack time. Then $s \geq 0.207q$

Proof : We examine two cases :

Case 1 : The execution of the last quantum of the request begins at no more than T_1 units of time after the arrival of the request, as illustrated in Fig. 2(a).

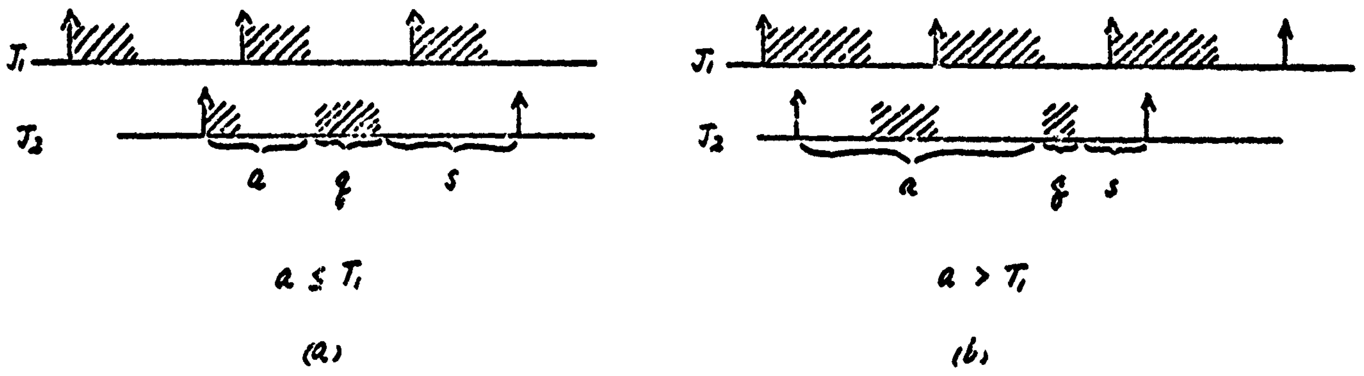


Fig. 2.

In this case, since $a \leq T_1$, during this a units of time atmost C_1 units of computation time was devoted to J_1. Thus we have

$$s \geq T_2 - (C_1 + C_2) = T_2\left(1 - \frac{C_1 + C_2}{T_2}\right)$$

$$\geq T_2\left[1 - \left(\frac{C_1}{T_1} + \frac{C_2}{T_2}\right)\right]$$

$$= T_2(1 - u)$$

$$= C_2\left(\frac{1-u}{u_2}\right)$$

$$\geq q\left(\frac{1-u}{u}\right) \geq q\left(\frac{1-u}{u}\right) \geq q\left(\frac{1-0.83}{0.83}\right) = 0.207q$$

Case 2 : The execution of the last quantum of the request begins at more than T_1 units of time after the arrival of the request, as illustrated in Fig. 2(b).

Let b denote C - q. We have

$$u = \frac{C_1}{T_1} + \frac{C_2}{T_2} = u_1 + \frac{b+q}{a+q+s}$$

or

$$b + q = (a + q + s)(u-u_1)$$

or

$$s(u-u_1) = q(1-u+u_1) + b - a(u-u_1) \qquad (6)$$

Consider a job that has a request period equal to a and computation time equal to b. Suppose we want to schedule the two jobs $(C_1, T_1)(b,a)$ according to the rate monotonic scheduling algorithm. According to Fig. 2(b), and Theorem 3, after J_1 was scheduled the total processor time available for J_2 in the time interval $[0, a]$ is less than or equal to b. Consequently, either the set $\{(C_1, T_1), (b, a)\}$ fully utilizes the processor or the set is not schedulable at all. In either case, according to Theorems 1 and 2, we have :

$$u_1 + \frac{b}{a} \geq 2(2^{1/2}-1) > u$$

That is

$$b - a(u - u_1) > 0$$

Consequently, (6) yields

$$s(u - u_1) \geq q(1 - u + u_1)$$

$$s \geq q\left(\frac{1-u_2}{u_2}\right) \geq 0.207q$$

We now have :

<u>Theorem 4</u> : Let $J_1, J_2, \ldots, J_n$ be n jobs scheduled according to the rate monotonic scheduling algorithm. Let q denote the length of the last quantum of time allocated to the first request. If $u < n(2^{1/n}-1)$ then the slack time of <u>any</u> request, is larger than or equal to 0.207 q.

Proof : According to Theorem 3, the slack time of the first request is less than or equal to the slack time of any request. Thus, it is sufficient to prove that the slack time of the first request is larger than or equal to 0.207 q.

The theorem is proved by induction on n. Lemma 3 provides the basis of induction, As to the induction step, we assume that the theorem is true for n -1 jobs. Let us use q and s to denote the length of the last quantum of time allocated to the first request of J_n and the slack time of this request. We consider two cases :

Case 1 : The execution of the last quantum of time allocated to request begins at no later than $t = T_{n-1}$ as illustrated in Fig.3(a).

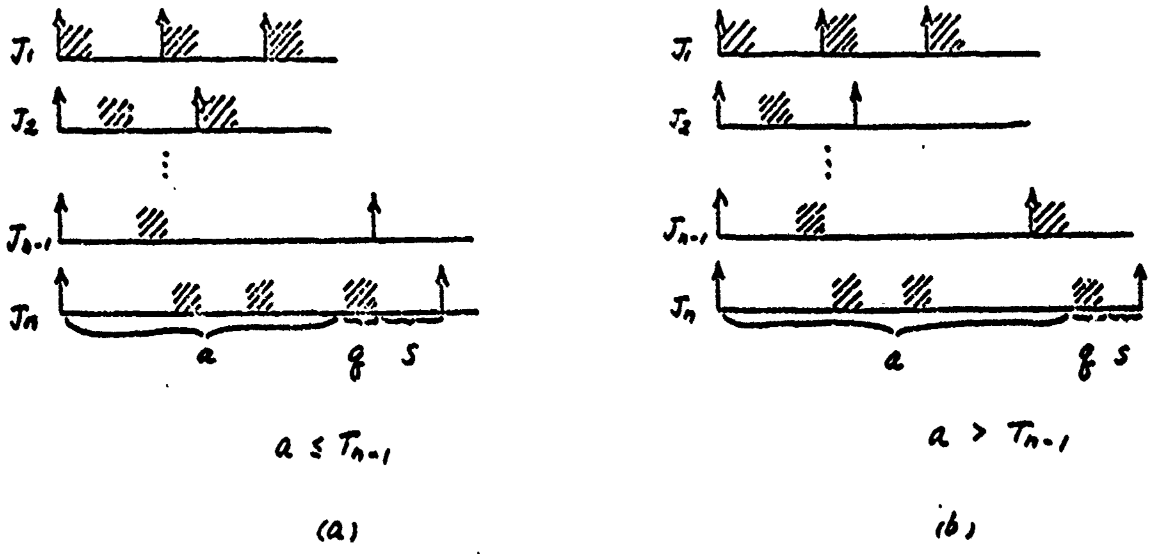


Fig.3.

Consider the set of n - 1 jobs $(C_1, T_1), (C_2, T_2), \ldots, (C_{n-2}, T_{n-2})$, $(C_{n-1} + C_n, T_n)$. According to the diagram in Fig.3(a) and Theorem 3, this set of n - 1 jobs is schedulable by the rate monotonic scheduling algorithm. Furthermore, the first request of the job $(C_{n-1} + C_n, T_n)$ will occupy the processor during the time intervals in which the first request of J_{n-1} and the first request of J_n occupy the processor when the jobs $J_1, J_2, \ldots, J_n$ were scheduled. Let q' and s' denote the length of the last quantum of time allocated and the slack time of the first request of the job $(C_{n-1} + C_n, T_n)$. Then according to the induction hypothesis

$$s' \ge 0.207q'$$

Howerver, since

$$s' = s \qquad \text{and} \qquad q' \ge q$$

we have

$$s \ge 0.207\, q$$

Case 2 : The execution of the last quantum of time allocated to the request begins at later than $t = T_{n-1}$ as illustrated in the diagram in Fig. 3(b). Let b denote $C_n - q$. We have

$$u = \sum_{i=1}^{n-1} u_i + \frac{C_n}{T_n} = \sum_{i=1}^{n-1} u_i + \frac{b + q}{a + q + s}$$

which can be written as

$$s(u - \sum_{i=1}^{n-1} u_i) = q(1 - u + \sum_{i=1}^{n-1} u_i) + b - a(u - \sum_{i=1}^{n-1} u_i) \qquad (7)$$

Consider the n jobs $(C_1, T_1), (C_2, T_2), \ldots (C_{n-1}, T_{n-1}), (b,a)$.

According to the diagram in Fig.3(b), this set of jobs fully utilizes the processor with respect to the rate monotonic scheduling algorithm. (The processor was never left idle within the time interval [0, a]. Because if this was not the case, the first request of J_n will be completed earlier). It follows that

$$\sum_{i=1}^{n-1} u_i + \frac{b}{a} > u$$

or

$$b - a(u - \sum_{i=1}^{n-1} u_i) > 0$$

It follows that (7) can be written as

$$s(u - \sum_{i=1}^{n-1} u_i) \geq q(1 - u + \sum_{i=1}^{n-1} u_i)$$

or

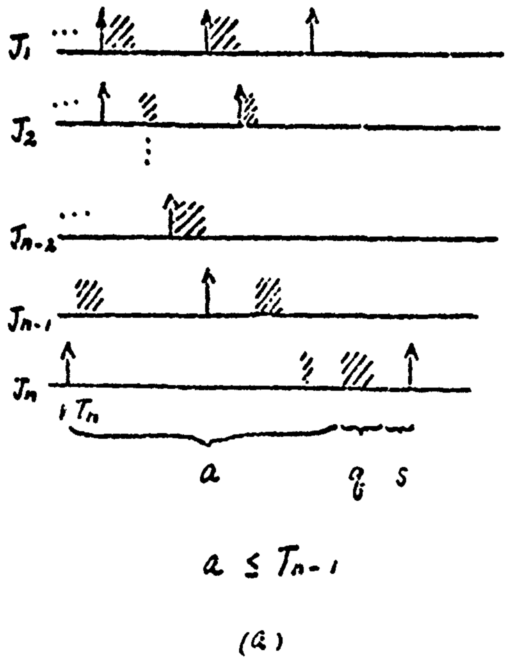
$$s \geq q\left(\frac{1 - u_n}{u_n}\right) \geq q\left(\frac{1-u}{u}\right) \geq 0.207\, q$$

We now have

Theorem 5 : Let $(C_1, T_1), (C_2, T_2), \ldots (C_n, T_n)$ be a set of n jobs with $u < n(2^{1/n}-1)$. Let $T_n \geq 2T_{n-1}$. For any request of J_n, let q and s denote the length of the last quantum of time allocated and the length of the slack time of any request. Then $s \geq 0.207\, q$.

Proof : The proof of the Theorem is carried out by induction on n. Lemma 3 provides the basis of induction. As to the induction step we consider two cases :

Case 1 : The execution of the last quantum of time allocated to the request begins at no more than T_{n-1} units of time after the arrival of the request as illustrated in Fig. 4(a).



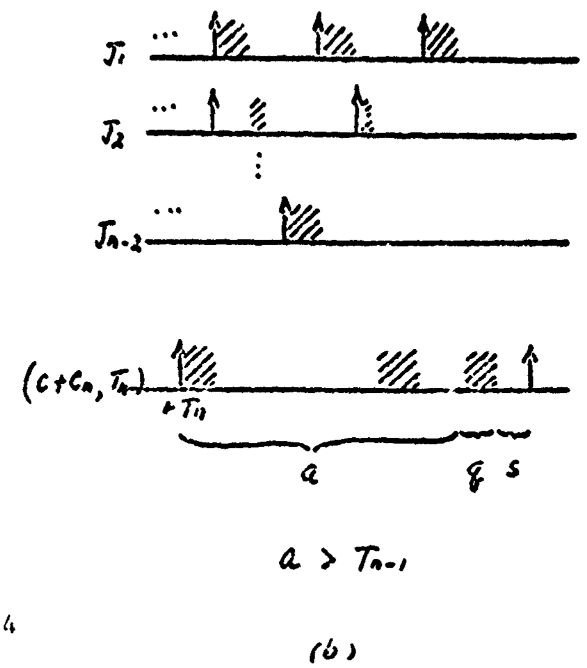


Fig. 4

Let C denote the total computation time allocated to J_{n-1} within the time interval $[rT_n, rT_n + a]$. Since $a \leq T_{n-1}$, we have $C \leq 2C_{n-1}$.
Now consider the n-1 jobs (C_1, T_1), (C_2, T_2), ... (C_{n-2}, T_{n-2}), $(C + C_n, T_n)$.
Since

$$\sum_{i=1}^{n-1} u_i + \frac{C + C_n}{T_n} \leq \sum_{i=1}^{n-2} u_i + \frac{2C_{n-1}}{T_n} + \frac{C_n}{T_n} \leq \sum_{i=1}^{n-2} u_i + \frac{2C_{n-1}}{2T_{n-1}} + \frac{C_n}{T_n} \leq u$$

these n-1 jobs are schedulable by the rate monotonic scheduling algorithm. Moreover, the diagram for the time interval $[rT_n, (r+1)T_n]$ will be that shown in Fig. 4(b). For there n-1 jobs, since $T_n \geq 2T_{n-2}$, by the induction hypothesis

$$s' \geq 0.207\, q'$$

Since

$$s' = s \qquad q' \geq q$$

we have

$$s \geq 0.207q$$

Case 2 : The execution of the last quantum of time allocated to the request begins at later than T_{n-1} units of time after the arrival of the request. The derivation of the bound for this case is similar to case 2 in the proof of Theorem 4, and will not be repeated here.

5. Remarks.

Aside from the results in Theorems 4 and 5 which give a lower bound of the slack time of a request as a function of q, it seems that slack time can be estimated in other ways, in particular, as a function of the utilization factor of a set of jobs. Intuitively, it is clear that for a set of jobs with a small utilization factor, the slack time of each request should become large. Our result only uses the utilization factor in an implicit way.

We conjecture that the condition $T_n \geq 2T_{n-1}$ in Theorem 5 is unnecessary.

REFERENCES

1. Dertouzos M.L., "Control robotics : the procedural control of physical processes", IFIP Proc., 1974, p. 807-813.
2. Labetoulle J., "Some theorems on real time scheduling", Computer architectures and networks, E. Gelenbe and R. Mahl eds., North-Holland Publ. Co., 1974, p. 285-298.
3. Liu C.L. and J.W. Layland, "Scheduling algorithms for multiprogramming in hard-real-time environment", JACM, Jan. 1973, p. 46-61.